REMARKS

Reconsideration of the application is requested.

Applicant acknowledges the Examiner's confirmation of receipt of applicant's certified copies of the priority documents for the German Patent Application 100 00 960.3, filed January 12, 2000 supporting the claim for priority under 35 U.S.C. § 119.

Claims 1-12 remain in the application. Claims 1 and 7 have been amended.

In "Specification" item 4 on page 2 of the above-identified Office Action, the Examiner objected to the title of the invention. The Examiner's suggested corrections have been made.

In "Specification" item 5 on page 2 of the above-identified Office Action, the Examiner objected to the abstract of the disclosure. The Examiner's suggested corrections have been made. To comply with 37 CFR § 1.72, a clean version of the replacement abstract is enclosed on a separate sheet with this amendment.

In "Claim Objections" item 7 on page 3 of the above-identified Office Action, the Examiner objected to claim 9. More specifically, the Examiner states:

> the claim language recites the language "a first multiplexer writing either the output signal of said adder or the addresses for program jumps or function calls into said program counter assigned to the active process." A "de-multiplexer" more generally performs the operation described in this language, as it is not selecting on out of the two inputs as a multiplexer conventionally does, but is rather selecting between two inputs to be output to one of many outputs.

Applicant respectfully traverses the objection.

The design of the branching control unit 11 of the instant application is illustrated in FIG. 3 and described in the respective description found on page 19, line 9. The output of adder 30 is fed to a first multiplexer 31. The other input of the first multiplexer 31 receives the Br_Ctr bus 36, configured to control jumps and function calls. The first multiplexer 31 either switches the signals supplied by the adder 30 or the signals supplied via the Br_Ctr bus 36 to one of two program counters 32 and 33. The first multiplexer 31 selects one of the two program counters 32 and 33 to write to based on the process number TNr supplied via a thread bus 37. A program counter is assigned to each of the active processes (page 19, lines 17-25).

Moreover, claim 9 states that the branching control unit includes "a first and a second multiplexer ... said first multiplexer writing either the output signal of said adder or the addresses for program jumps or function calls into said program counter assigned to the active process". As the first multiplexer 31 either switches the output signals supplied by the adder 30 or the address signals supplied by the Br_Ctr bus 36 to the active program counter 32 or 33, the first multiplexer 31 is therefore described and being used properly.

Similarly, the second multiplexer 34 receives the output of the program counters 32 and 33 and switches output based on the process number TNr supplied via a thread bus 37.

Applicant believes the language of claim 9 accurately reflects the appropriate hardware component and respectfully requests the withdrawal of the Examiner's related objection.

In "Claim Rejections - 35 USC § 112" on page 4 of the above-identified Office Action, claims 1 and 7 have been rejected as being indefinite under 35 U.S.C. § 112, second paragraph.

More specifically, in regards to claim 1 on page 4 of the above-identified Office Action, the Examiner states that it

is unclear whether the "multiplicity of bundles" are included within the compiled program or whether there is merely information on the multiplicity of bundles. Applicant has amended claim 1 to clarify that the compiled program includes a multiplicity of bundles.

Additionally, in regards to claim 1 on page 4 of the above-identified Office Action, the Examiner states that it is unclear whether the program flow control unit is controlling the branch control unit or whether the bundles are fetched from the branch control unit. Applicant has amended claim 1 to clarify that the program flow control unit controls at least three items: the fetching of bundles from the program memory, the branching control unit, and the output of instructions.

In regards to claim 7, the Examiner, on page 5 of the above-identified Office Action, states that it is unclear whether the program flow control unit, the execution units, or both units execute bundles of instructions. The applicant has amended to claim 7 to clarify that the data-processing device includes "a plurality of instruction-execution units connected to said program flow control unit and configured to execute the instructions of one or more bundles in parallel."

Support for these changes may be found, among other places, on page 14 of the specification of the instant application.

It is accordingly believed that the specification and the claims meet the requirements of 35 U.S.C. § 112, second paragraph. The above-noted changes to the claims are provided solely for clarification or cosmetic reasons. The changes are neither provided for overcoming the prior art nor do they narrow the scope of the claim for any reason related to the statutory requirements for a patent.

In "Claim Rejections - 35 USC § 102" item 14 on page 6 of the above-identified Office Action, claims 1-8 and 10 have been rejected as being fully anticipated by U.S. Patent No. 5,941,983 to *Gupta, et al.* (hereinafter **GUPTA**) under 35 U.S.C. § 102(a).

In "Claim Rejections - 35 USC § 103" item 27 on page 11 of the above-identified Office Action, claim 11 has been rejected as being obvious over **GUPTA** in view of U.S. Patent No. 5,742,782 to *Ito, et al.* (hereinafter **ITO**) under 35 U.S.C. § 103(a).

In "Claim Rejections - 35 USC § 103" item 30 on page 11 of the above-identified Office Action, claim 12 has been

rejected as being obvious over **GUPTA** in view of U.S. Patent No. 6,404,752 to *Allen, Jr., et al.* (hereinafter **ALLEN**) under 35 U.S.C. § 103(a).

As will be explained below, it is believed that the claims were patentable over the cited art (**GUPTA**, **ITO**, and **ALLEN**) in their original form, however, for purposes of clarification claim 1 has been amended to further emphasize that the bundles containing the instructions are "to be processed in parallel" as recited in claim 1 of the instant application.

Before discussing the prior art in detail, it is believed that a brief review of the invention as claimed, would be helpful. Claim 1 calls for, *inter alia*, a data-processing device **for processing in parallel a plurality of independent processes** including:

> a program memory having stored therein at least one compiled program **with a multiplicity N of independent processes**, the compiled program including **information on parallelism** and including **a multiplicity of bundles** with a plurality of instructions of a process, and
>
> **a program flow control unit** connected to a branching control unit, the program flow control unit **controlling a fetching of bundles to be processed in parallel** from the program memory, controlling **the branching control unit**, and controlling **an output of instructions to be processed in parallel** in dependence on information contained in the instructions and included in a compiling time of the program.

The **GUPTA** reference discloses a method for executing instructions out-of-order to improve performance of a

processor. The method includes compiling the instructions of ***a single program*** into separate queues along with various encoded dependencies between instructions in the different queues. The processor then issues instructions from each of these queues independently, except that the processor enforces the encoded dependencies among instructions from different queues.

However, during execution of the program, the processor in **GUPTA** "issues instructions from each queue **in sequential order**" (col. 6, lines 15 and 16) and not "in parallel" as recited in claim 1 of the instant application. Moreover, **GUPTA** further discloses that the processor can issue instructions from each queue independently and as fast as a functional unit can handle new instructions **as long as the dependencies are satisfied** (Col. 6, lines 26 to 29).

In contrast, the present invention as claimed in independent claim 1, provides a data-processing device wherein the data-processing device or processor can carry out/process ***at least two independent processes*** (or threads) in parallel.

Accordingly, the claimed invention benefits from the fact that the dependencies of data and instructions in independent processes running in parallel are typically significantly

smaller than in an individual program. For example, the single sequential program flow as described in **GUPTA** (see *e.g.*, col. 6, lines 15-16) has more dependencies of data and instructions for parallel processing than when at least two independent processes are processed in parallel as described in the instant application. More specifically, in the instant application the instructions, which are to be processed in parallel, are fetched from a program memory according to a clock cycle and each individual parallel process is assigned a priority.

Clearly, **GUPTA** does not show a "data-processing device **for processing in parallel a plurality of independent processes**" as recited in claim 1 of the instant application. Nor does GUPTA teach or suggest "**a program flow control unit** … controlling a **fetching of bundles to be processed in parallel from said program memory**, … and **controlling an output of instructions to be processed in parallel** in dependence on information contained in the instructions and included in a compiling time of the program" as recited in claim 1 of the instant application.

Moreover, as will be shown below, neither **ITO** nor **ALLEN** compensate for the deficiencies previously shown in **GUPTA**. To establish a *prima facie* case of obviousness MPEP 2142

indicates that three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations.

As such, a *prima facie* case of obviousness for claim 11 must provide a motivation to combine **ITO** with **GUPTA** AND the proposed combination of **GUPTA** and **ITO** must have a reasonable expectation of success AND the proposed combination must teach all of the claim limitations. In addition to having no motivation to combine parallelism directed towards "out of order execution" for a single process as described in **GUPTA** with a processor for executing "a plurality of VLIW threads in parallel" in **ITO**, the proposed combination does not teach all of the claim limitations as demonstrated below.

In addition to the distinctions previously discussed over **GUPTA**, the Examiner correctly notes in item 28 on page 11 that claim 11 also states that "a process is called by assigning a process number, a priority, and a memory address of a starting point of the process in the program memory"

which is not taught or suggested by **GUPTA**. As indicated above, the Examiner asserts that these deficiencies may be found in **ITO**, applicants respectfully traverse.

The **ITO** reference discloses a processing apparatus for executing a plurality of **VLIW threads** in parallel. The processing apparatus simultaneously executes multiple threads of long instructions, where each thread is made up of multiple operational instructions. The gain by virtue of using ILP is, however, restricted by the inherent dependencies of the data operations and control operations. In order to avoid such dependencies, complex preprocessing - for example taking into account data and control operation dependencies during the actual programming - is necessary and this in turn makes the entire development process more expensive.

ITO does not assign "a process number, a priority, and a memory address ... in the program memory" as recited in claim 11 of the instant application. Rather **ITO** develops these signals during execution and changes the cited signals during the course of execution. For example, FIG. 13 indicates that the TNC/Signal155 or alleged "priority" changes in block (6) and block (4). Thus, TNC appears to be more like a counter or queue position than a priority. In contrast, FIG. 4 of

the instant application shows the state diagram of the flow control unit 10, in which the priority of process A is higher than the priority of process B.

Even assuming, *arguendo*, that the "STN" is a process number and that "signal155" or TNC does represent a priority, **ITO** does not **call a process** "by assigning a process number, a priority, and a memory address ... in the program memory" as recited in claim 11 of the instant application.

Clearly, **ITO** does not show "**assigning** a process number, a **priority**, and a memory address of a starting point of the process in the program memory" as recited in claim 11 of the instant application. Nor does **ITO** teach or suggest "**a process is called by assigning** a process number, a priority, and a memory address of a starting point of the process in the program memory" as recited in claim 11.

With regards to claim 12, a *prima facie* case of obviousness must provide a motivation to combine **ALLEN** with **GUPTA** AND the proposed combination of **GUPTA** and **ALLEN** must have a reasonable expectation of success AND the proposed combination must teach all of the claim limitations.

In addition to the distinctions previously discussed over **GUPTA**, the Examiner correctly notes in item 31 on page 12 that claim 12 also states that "said data-processing device is a network processor for processing layer 1 to 7 of protocol stacks" which is not explicitly taught by **GUPTA**. As previously discussed, the Examiner asserts that these deficiencies may allegedly be found in **ALLEN**, applicants respectfully traverse.

The **ALLEN** reference discloses a network switch in which data flow handling and flexibility is enhanced by a network processor capable of cooperating with other elements, such as an optional switching fabric device, to execute instructions that direct the flow of data within a network. The network processor includes a plurality of cooperating interface processors and various peripheral elements **formed on a semiconductor substrate**. **ALLEN** also discloses, "network processors can increase bandwidth and solve latency problems in a broad range of application by allowing networking tasks previously handled in software to be executed in hardware" (col. 2, lines 64-67). However, the combination of **ALLEN** and **GUPTA** would require using **multiple network processors**, where each processor may process a single process in parallel. Unfortunately, this would require a tremendous amount of processing overhead to coordinate parallel processing and

indicates that there is not a reasonable expectation of success for the proposed combination.

In contrast, the configuration described in claim 12 of the instant application handles multiple processes in parallel in each processor and could easily be used in a network processor configuration, because there is no need to coordinate the dispersal between the units.

Clearly, **ALLEN** does not show "**A data-processing device** for processing in parallel a plurality of independent processes" as recited in claim 12 of the instant application. Nor does **ALLEN** teach or suggest "a network processor for processing layer 1 to 7 of protocol stacks in applications including LAN, ATM switches, IP routers, and frame relays based on a system selected from the group consisting of DSL, Ethernet, and cable modems" as recited in claim 12.

It is accordingly believed to be clear that none of the references (**GUPTA**, **ITO**, and **ALLEN**), whether taken alone or in any combination, either show or suggest the features of claim 1. Claim 1 is, therefore, believed to be patentable over the art. The dependent claims are believed to be patentable as well because they all are ultimately dependent on claim 1.

In view of the foregoing, reconsideration and allowance of claims 1-12 are solicited.

In the event the Examiner should still find any of the claims to be unpatentable, counsel would appreciate receiving a telephone call at (954) 885-1100 so that, if possible, patentable language can be worked out.

If an extension of time is required for this paper, petition for extension is herewith made. Kindly charge the extension fee and any other fees that might be due with respect to Sections 1.16 and 1.17 to the Deposit Account of Lerner and Greenberg, P.A., No. 12-1099.

Respectfully submitted,

Kyle H. Flindt
Reg. No. 42,539

For Applicant

KHF:cgm

April 13, 2004

Lerner and Greenberg, P.A.
P.O. Box 2480
Hollywood, Florida 33022-2480
Tel.: (954) 925-1100
Fax: (954) 925-1101

ABSTRACT OF THE DISCLOSURE

A data-processing device, in particular a network processor for processing layer 1 to 7 of protocol stacks in applications such as LAN, ATM switches, IP routers or frame relays which are based on DSL, Ethernet or cable modems is enabled to process at least two independent processes in parallel. The processor has instruction buffers, instruction decoders, and instruction-execution units corresponding to the number of processes to be processed in parallel. A program flow control unit essentially controls the parallel processing, by controlling the fetching of bundles from a program memory and a branching control unit, and also by controlling an output of executable instructions.